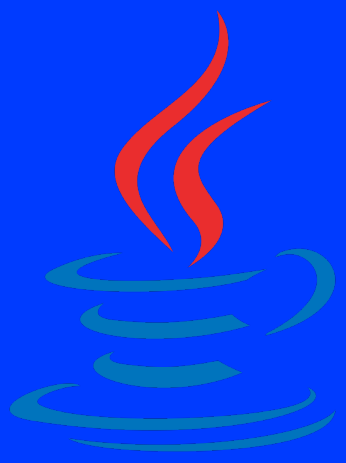
{EPITECH}

JAVA SEMINAR_

< DAY 03 - PACKAGES />

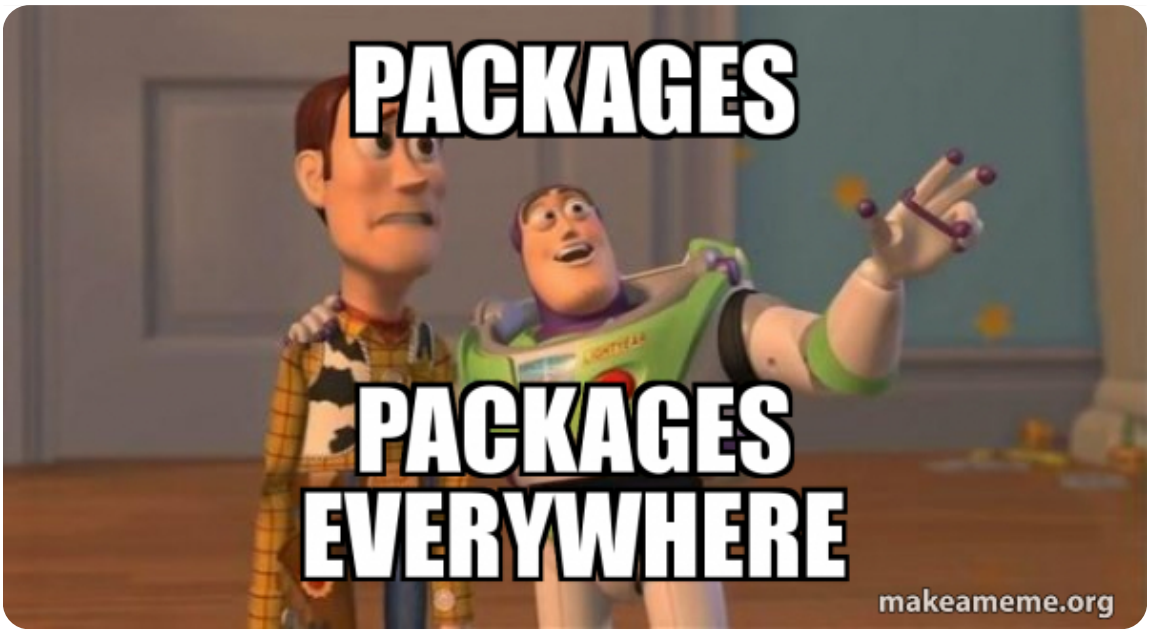


JAVA SEMINAR

Let's keep going deeper into OOP!

You will today keep using all yesterday's concepts, and you will also discover a few new things:

- ✓ static keyword
- ✓ packages



⚠️ Unless otherwise specified, all messages must be followed by a newline.

Exercise 01

Delivery: `./ex_01/Mars.java`

Create a new class, named `Mars`, that has an `id` attribute, a getter (**getId**), but no setter.

Create it so that the first instance id is 0, the second instance id is 1, ...

static?!

```java
public class Example {
    public static void main(String[] args) {
        Mars rocks = new Mars();
        Mars lite  = new Mars();
        Mars snack = new Mars();

        System.out.println(rocks.getId());
        System.out.println(lite.getId());
        System.out.println(snack.getId());
    }
}
```

Terminal

```
$> java Example
0
1
2
```

From now on, the files and classes will be reused and expanded across exercises.

Exercise 02

Delivery: `./Astronaut.java`

Create a new `Astronaut` class, with the following attributes:

- ✓ `name`: a string describing the name of the Astronaut ;
- ✓ `snacks`: an integer describing the number of snacks your Astronaut possess ;
- ✓ `destination`: a string describing the destination of the Astronaut ;
- ✓ `id`: an integer describing the id of the Astronaut.

> ⚠ The name must be passed during the creation of the Astronaut.
> It is mandatory.

His `snack` will be initialized to 0 and his `destination` to `null`.

The id is unique, starts from 0 and is incremented for each new Astronaut created.

Also, every Astronaut being created must display `[name] ready for launch!`.

All these attributes must have an associated getter, but no setter.

```java
public class Example {
    public static void main(String[] args) {
        Astronaut mutta = new Astronaut("Mutta");
        Astronaut hibito = new Astronaut("Hibito");

        System.out.println(mutta.getId());
        System.out.println(hibito.getId());
    }
}
```

Terminal

```
$> java Example
Mutta ready for launch!
Hibito ready for launch!
0
1
```

Exercise 03

Delivery: `./chocolate/Mars.java`, `./planet/Mars.java`

Copy your `Mars` class from the first exercise, without changing it.
Create another `Mars` class representing the planet.

In order to differentiate between the two Mars, put:

- ✓ the first one in a package called `chocolate` ;
- ✓ the second one in a package called `planet`.

Add an attribute `landingSite` of type String to the planet, and its getter.

When creating a planet Mars instance, specify the name of the landing site in the constructor.

```java
import chocolate.*;
import planet.*;

public class Example {
    public static void main(String[] args) {
        chocolate.Mars snack = new chocolate.Mars();
        planet.Mars rock = new planet.Mars("Viking 1");

        System.out.println(snack.getId());
        System.out.println(rock.getLandingSite());
    }
}
```

Terminal

```
$> java Example
0
Viking 1
```

Exercise 04

Delivery: `./Astronaut.java`, `./chocolate/Mars.java`, `./planet/Mars.java`

> `[Name]` should be replaced by the name of the Astronaut while displayed:
> For instance, `[Name]: Nothing to do.` becomes something like `Mutta: Nothing to do.`
> The same goes for other values inside brackets.

It is time for your Astronaut to start working!
Create a new method `doActions` taking an optional parameter.

This method displays:

- ✓ `[Name]: Nothing to do.`, if no parameter is given ;
- ✓ `[Name]: Started a mission!`, if the parameter is a `planet.Mars` ;
- ✓ `[Name]: Thanks for this Mars number [Mars id]`, if the parameter is a `chocolate.Mars` ;

Depending on the case, you will need to:

- ✓ store the planet landing site as your Astronaut's new `destination` ;
- ✓ or increment his `snacks` attribute by one.

After each previous sentence, if the astronaut has no destination, it will also display:
`[Name]: I may have done nothing, but I have [x] Mars to eat at least!`

Exercise 05

Delivery: `./planet/moon/Phobos.java`, `./planet/Mars.java`

Create a `Phobos` class in the `Phobos.java` file.

This class must be in a `moon` package, which is, itself, defined in the `planet` package.
Your `Phobos` class must have a private attribute named `mars` with a getter (`getMars`), but no setter.
This attribute must be specified upon creation.

It is a representation of a `planet.Mars` followed by a `landingSite` attribute.
This attribute is also stored in the class and has it's own getter.

During its creation, it displays:

- ✓ `Phobos placed in orbit.`, if it correctly received a `planet.Mars` ;
- ✓ `No planet given.` otherwise.

```java
import planet.*;

public class Example {
    public static void main(String[] args) {
        planet.Mars titi = new planet.Mars("Here and there");
        planet.Mars toto = new planet.Mars("Up");
        planet.moon.Phobos phobos1 = new planet.moon.Phobos(titi, "Alpha 3");
        planet.moon.Phobos phobos2 = new planet.moon.Phobos(toto, "Beta 1");
        System.out.println(phobos1.getLandingSite());
    }
}
```

Terminal

```
$> java Example
Phobos placed in orbit.
Phobos placed in orbit.
Alpha 3
```

Exercise 06

Delivery: `./planet/Mars.java`, `./Astronaut.java`, `./Team.java`

Create a new `Team` class that represents a team of astronaut.
Its constructor must take the team name as parameter.
Create a getter (`getName`), but no setter for this attribute.
Create a few methods to manipulate your team:

- ✓ `add` takes an `Astronaut` as parameter and add it to the team ;
- ✓ `remove` takes an `Astronaut` as parameter and removes it from the team ;
- ✓ `countMembers` returns the number of `Astronaut` currently on your team ;
- ✓ `showMembers` displays the members that are on the team;
  - like this `[Team name]: [Astronaut 1] on mission, [Astronaut 2] on standby.`
  - `on mission` is displayed if the Astronaut is currently on a mission.
  - otherwise, `on standby` is displayed.
  - if no member is in the team, don't display anything.

```java
import planet.*;

public class Example {
    public static void main(String[] args) {
        Astronaut mutta = new Astronaut("Mutta");
        Astronaut hibito = new Astronaut("Hibito");
        Astronaut serika = new Astronaut("Serika");
        Team spaceBro = new Team("SpaceBrothers");
        spaceBro.add(mutta);
        spaceBro.add(hibito);
        spaceBro.add(serika);

        System.out.println(spaceBro.countMembers());

        planet.Mars titi = new planet.Mars("Hill");
        mutta.doActions(titi);
        spaceBro.showMembers();
        spaceBro.remove(hibito);

        System.out.println(spaceBro.countMembers());
    }
}
```

Terminal

```
$> java Example
Mutta ready for launch!
Hibito ready for launch!
Serika ready for launch!
3
Mutta: Started a mission!
SpaceBrothers: Mutta on mission, Hibito on standby, Serika on standby.
2
```

Exercise 07

Delivery: `./planet/moon/Phobos.java`, `./chocolate/Mars.java`, `./planet/Mars.java`, `./Astronaut.java`, `./Team.java`

Add a new method to your Team, `doActions`.

This method calls all of the Team's Astronaut's `doActions` with the received parameter.
If no parameter is received, it displays `[Team name]: Nothing to do`.

> Display it only once.

> If `chocolate.Mars` is received as parameter, we will admit that the team share the chocolate but it still count as a full chocolate for each astronauts.

Now that your Astronauts have more experience, they can also go on a mission to Phobos.
You will need to modify your Astronaut class.

v 3.4.1